Journal of Theoretical and Applied Information Technology
31st January 2021. Vol.99. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DISCOVERY OF MAXIMAL FREQUENT ITEMSET USING PRIME ALGORITHM

[1]R.SMEETA MARY, [2]DR.K.PERUMAL

[1]Assistant Professor, Department of Computer Applications, Fatima College, Madurai, India

[2]Professor, Department of Computer Applications, Madurai Kamaraj University, Madurai, India

E-mail: [1]smeetamaryr@gmail.com, [2]perumalmala@gmail.com

ABSTRACT

Data mining is the technique of discovering the new patterns in large data sets with reference to various methods at the intersection of statistics, machine learning and database systems. Computer science and statistics are the interdisciplinary subfield of data mining. The overall goal of data mining is to extort information from a data set and renew or reframe the information into a structure. Association rule is a research area in the field of data searching for frequent and using the criteria support to find frequently the items appear in the data and confidence to identify the most important relationships. In focus of this paper is to find maximal frequent itemset using a new algorithm called maximal Frequent Itemset using Prime algorithm. Most of the association rule algorithms are used to find the minimal frequent item set, and then with the help minimal frequent item set derive the maximal frequent item set. But it consumes lot of time. So to overcome this problem a new approach Maximal Frequent Itemset using Prime algorithm is proposed to find the maximal frequent item set directly. The proposed method is efficient in finding the maximal frequent item set

Keywords: *Data Mining (DM), Association Rules (AR), Frequent Itemset (FIS), Maximal Frequent Itemset using Prime algorithm (MFIPA)*

1. INTRODUCTION

Data mining is a logical process which is used to search large amount of data to find useful data. Finding out the previously unknown data is the goal of this technique. There are various steps involved such as Exploration, Pattern identification and Deployment. In exploration the data is cleaned and transformed into new form. The important variables and nature of the data based on the problem are found out. Once the data is explored using pattern identification, then it is refined and defined for the certain variables. It makes the best prediction by identifying and choosing the patterns. The last step deployment is finding out the desired outcome by deploying the patterns.

In this process the first step is Data cleaning in which noise and inconsistent data is removed and multiple data sources are combined together. In the data selection the data that are relevant for the analysis task are retrieved from the database. Data mining algorithms are used to find the extract data patterns. Some interesting measures are used for the identification of the data patterns. Using many knowledge representation techniques, knowledge is shared to the user.

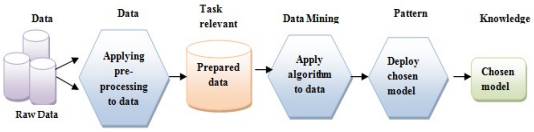

Figure 1: Process of Data Mining

At a basic level, association rule mining involves various learning models to analyze data for patterns, or co-occurrence, in a database. Association rule mining is divided into two parts that are antecedent and a consequent. An antecedent is an item which is found within the data. A consequent is an item which is found in combination with the antecedent. By using both antecedent and a consequent the association rules are created for searching the data. Hence Support and Confidence plays a vital role in identifying the relationships. Support indicates how frequently the item appears in the data and confidence indicates how many times the condition is true. In 2000, J. C. Fernando Berzal

promoted an efficient method for association rule mining in relational databases [2]. This algorithm is used to find the frequent sets whose support will always be greater than minimum support. The strong association rule generates frequent item set that satisfy minimum confidence and support. But it has various disadvantages that this algorithm needs more complexity and scans the transaction database many times. In current research these issues are the disadvantages. In this paper, faster and more efficient algorithm PRIMA is promoted.

2. THEORETICAL CONCEPTS

Association Rule: Association rule mining is a method to discover correlations, frequent patterns, associations, or causal structures from data sets found in a variety of kinds of databases such as transactional databases, relational databases and other forms of data repositories.

Given a set of transactions, association rule mining intend to find the rules which enable us to expect the occurrence of a specific item based on the occurrences of the other items in the transaction.

Frequent itemsets: A set of items that materialize in many transactions is understood to be "frequent". Frequent itemset is that itemset whose support must be greater than or equal to a minimum support threshold. Frequent mining is creation of association rules on or after a transaction

3. RELATED WORK

There are number of attempt to find maximal frequent itemsets. In [3], they focused on parallel algorithms which are used to discover either maximal itemsets or frequent or closed frequent in order to solve the performance worsening, load balancing and scalability dispute of sequential algorithm. In [4], AIS algorithm was first proposed by Agarwal and Swami for mining association rule. In AIS algorithm the databases are scanned many times to get the frequent itemsets. During the first pass over the database the support count of each individual item was calculated. Item whose count is less than its minimum value are eliminated from the list of items with respect to the threshold of support count. In the second pass over the database, the support count of those candidate 2 itemsets are accumulated and checked against the support threshold. AIS algorithm has competence problems so some modifications have been introduced to give estimation for candidate itemsets that have no hope to be large. In [5], Apriori algorithm, the name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemsets properties. Apriori uses an iterative approach known as level wise search, where k itemsets are used to explore k+1 itemsets. First the frequent 1 itemsets is found, this is denoted by L1, frequent2 itemset L2 and so on. Two step processes is used to find LK, one is the join step and other is prune step to find the frequent itemset. In [6], An MFIMiner algorithm uses breadth-first search with resourceful pruning method that expertly mines both long and short maximal frequent itemsets. In [7], this search is used only for maintaining and updating a new data structure of the maximum frequent candidate set. It is used to prune candidates in the bottom up search. The very important distinctive of this algorithm is that it does not require any examination of every frequent itemset explicitly. So it performs well even when some maximal frequent itemsets are long.In [8], GenMax algorithm, which uses backtrack search method for mining maximal frequent itemset. It uses various optimizations methods to prune the search space. Progressive focusing technique is used to perform maximality checking, and different set of propagation to perform fast frequency computation. It is found that GenMax to be a highly efficient method to mine the exact set of maximal patterns. In [9] a one-pass algorithm called Data Stream Mining for Maximal Frequent Itemsets which gets the set of all maximal frequent itemsets over data streams. A data structure called summary frequent itemset forest is developed for incrementing and maintaining the information of obtaining maximal frequent itemsets in the stream.

3.1 Maximal Frequent Itemset using Prime Algorithm

Most of the algorithms used for mining maximal frequent itemsets perform fairly well when the length of the maximal frequent itemset is small. However, performance degrades when the length of the maximal frequent itemset is large. The Maximal Frequent Itemset Using Prime Algorithm proposes a new approach for mining maximal frequent itemsets. It reduces the complexity by counting the items in the horizontal method and arranging the elements in the descending order for generating maximal itemsets. The search starts by assigning the prime

value to all the transactions. The values are being assigned from 1-itemset and proceeds upto n-itemsets. According to the preference the values are subtracted from the total count from n itemsets and proceeds upto 1-itemset. This search identifies the maximal frequent itemsets by examining its candidates that leads to 0. The search using this algorithm moves one-level up during a single pass whereas top-down search or bottom up approach moves many levels down during a single pass.

4. PROBLEM DESCRIPTION

4.1 Procedure

Step 1: Collect the input in individual list and put them into Hashset (Original InputSet).

Step 2: Count the individual itemcount in horizontal manner, and put it in HashSet (CountSet).

Step 3: Sort the Individual items in the HashSet by values in descending order.

Step 4: Assign consecutive prime numbers to individual items, SubstituteValSet.

Step 5: Apply prime numbers to all the individual items present in the transaction and find the total, SubstituteVal.

Step 6: while (SubstituteVal>0), Find the index of each element in an arraylist, sortedByCount and find the SubstituteValueSet of that particular element.

Step 7: while (HashSet(CountSet)>0), Subtract SubstituteValSet from SubstituteVal of all the transactions and store the value in HashSet(Qualified ItemCount) and add the element name, freqelement[].

Step 8:if any of the transactionnHashSet(Qualified ItemCount)=0 then stop the process else repeat step5 and 6.

4.2 Procedure explained with the example

Here 6 transactions and 5 items A, C, D, T and W to find maximal frequent itemset shown in below table.

Step 1: Collect the input in individual list and put them into Hashset(Original InputSet)

Table 1. Original Dataset

Tid	Titems				
T1		B	C	D	E
T2	A		C	D	E
T3		B	C	D	E
T4	A		C	D	E
T5	A	B	C	D	
T6		B	C		E
…	…	…	…	…	…
T294	A	B	C	D	E

Step 2: Count the individual item count presents in all transaction in horizontal manner and put in a two-dimensional array, HashSet. Once item frequencies are determined the algorithm will prepare the ignore list which includes a set of item not be considered in the further analysis.

Table 2. Individual Item Count

Titems	Count
A	198
B	213
C	294
D	175
E	270

Step 3: Sort the Individual items in the HashSet by values in descending order.

Table 3. Arrange in descending order

Titems	Count
C	294
E	270
B	213
A	198
D	175

Step 4: Assign consecutive prime numbers to each item in the dataset except the ones in the ignore list. Prime number assigning is done with respect to items frequency, SubstituteValSet.

Table 4. Substitute prime values

Titems	Prime values
C	2
E	3
B	5
A	7
D	11

Step 5: Apply prime numbers to all the individual items present in the transaction and find the total, SubstituteVal.

Table 5. Apply prime numbers

Titems					Total Count
	5	2	11	3	21
7		2	11	3	23
	5	2	11	3	21
7		2	11	3	23
7	5	2	11		25
	5	2		3	10
…	…	…	…	…	…
7	5	2	11	3	28

Table 6. Process the data

Tra	Tot	C	CE	CEB	CWBA
T1	21	19	16	11	4
T2	23	21	18	13	6
T3	21	19	16	11	4
T4	23	21	18	13	6
T5	25	23	20	15	8
T6	10	8	5	0	-
…	…	…	…	…	…
T294	28	26	23	18	

Step 6: While (SubstituteVal>0), Find the index of each element in an arraylist, sortedByCount and find the SubstituteValueSet of that particular element.

Step 7: While (HashSet(CountSet)>0), Subtract SubstituteValSet from SubstituteVal of all the transactions and store the value in HashSet(Qualified ItemCount) and add the element name, freqelement[].

Step 8: if any of the transaction HashSet(Qualified ItemCount) =0 then stop the process else repeat step 5 and 6. Here CEB,CEBA are considered as the maximal frequent item set.

Step 9: Check the support with the frequentelements. If suppose the support is 196 then CEB is displayed.

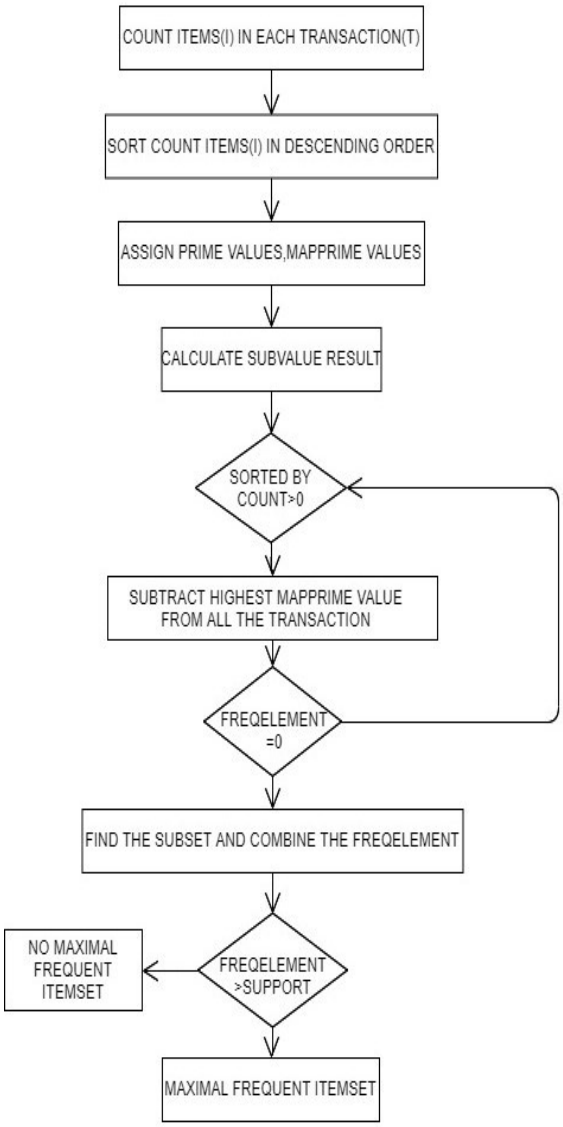

Figure 2: Procedure to find maximal frequent itemsets

4.3 MFIPA

Algorithm –MFIPA
Input: dataset D, support S
Output: Maximal Frequent Itemsets MFI
MFIPA (Dataset D)
BEGIN
1. Read and store the entire preprocessed dataset.
2. Traverse horizontally through all the elements in the list and generate the sum of

each item in respect to their column name, colList
3. Reorder the colList items in descending order.
4. Generate prime numbers for the each column items.

```
PrimeList = GetPrimeNumbers(colList);
```

5. If PrimeList is null: continue;
6. Substitute the prime numbers from the PrimeList to the respective colList items.

```
rowTotalList = []
for i = 0,…,len(colList) {
    initialize sum as 0
      for j=0,…,len(colList(i))
          sum = sum + colList(i)[j];
          rowTotalList.add(sum)
```

7. For primenumber ε PrimeList // PrimeList-> set of prime numbers

```
    get Colname of primenumber
    get the sum of Colname from colList
    if sum is unique
      resultList= from each rowTotalList value    subtract the primenumber
      CurColname.add(Colname)
      Adding the count of Colname as freqcolumn//for checking the support
      if resultList.contains(0)
          Add to FI(CurColname)
          break;
          else
      get Colnames of duplicatedsum from colList
    for i=0,…,count(Colnames):
      get primenumber of Colnames[i]
      resultList= from each rowTotalList value subtract the primenumber
        Adding the count of Colname as freqcolumn//for checking the support
          for each index, j ε resultList
            if j equals 0:
              pass
              else:
              rounds=count(remaining items in the resultList)
                result=[ ]
                  for k=0,…,rounds:
                get the consecutive primenumbers for remaining items
              result.add(subtract primenumber from the value
      until it reaches 0 or below))
          Adding the count of Colname as freqcolumn//for checking the support
END
```

GetPrimeNumbers
Input: colList
Output: A total number of Prime Numbers according to the count of colList items

```
GeneratePrimeNumbers(colList)
BEGIN
count = len(colList); //count will contain total number of columns
Initialize PrimeList as null
Input N and count
While N is smaller than count
    Initialize I to 2
        While I is smaller than N
        If  N is divisible by I
            skip loop
        Increment the value of I
        If  N is equal to I
        PrimeList.add(N)
        Increment the value of N
return PrimeList
END
```

GenerateCandidates
Input: Frequent Itemset, Candidate set
Output: Exact Candidates of Frequent Itemset.

```
GenerateCandidates(frequent,candidate
BEGIN
cand=null;
If(Columnfreq) ≥ support))
cand.add(item(Columnfreq);
return (cand);
END
```

5 EXPERIMENTAL ANALYSIS

To evaluate the performance of the proposed MFIPA, it is implemented in Java on a 2.10 GHZ Intel Pentium Dual Core with 4.00 GB of main memory. The performance of the proposed MFIPA is compared with Java version of GenMax and Apriori. Timing in the figure is depends on the total time comprise of all preprocessing costs.

Figure. 3 and Figure.4 shows the results of the new proposed method. This method takes less time to find maximal frequent item set. Figure. 3 consists of 6 transactions of 23 items as input, in which three transactions have frequent items and the values are similar.

Another example, Figure. 4 consists of 294 transactions of 12 items as input, here three items has different frequent items of these only one item has the maximum count as 6 taken from [10]. Here the count value is greater than all the remaining values, then that will be the Maximal frequent itemset.

```
Current Active Sheets Index: 0
Total number of Rows: 294
Maximum Data entered Column: 5
Input Data: [, B, C, D, E, A, , C, D, E, , B, C, D, E, A, , C, D, E
Items: [A, B, C, D, E]
Individual Items Count: {A=198, B=213, C=294, D=175, E=270}
Individual Items Count(Sorted): {C=294, E=270, B=213, A=198, D=175}
Assigned prime number to items: {C=2, E=3, B=5, A=7, D=11}
Item Row count: [21, 23, 21, 23, 25, 10]


Individual Items calculated Value:
**********************************
Item: C : [19, 21, 19, 21, 23, 8]
Item: E : [16, 18, 16, 18, 20, 5]
Item: B : [11, 13, 11, 13, 15, 0]
Item: A : [4, 6, 4, 6, 8, -7]
Item: D : [-7, -5, -7, -5, -3, -18]

Frequent Items:
***************
C -> E -> B
C -> E -> A
C -> E -> D
----------------------------------
CEB : 196 times
CED : 159 times
CEA : 180 times
Maximum frequent items Count: 196

********** FREQUENT ITEMS **************
CEB:196
****************************************
----------------------------------
1732694383
```

Figure3: Procedure to find maximal frequent itemsets

```
Total number of Rows: 6
Maximum Data entered Column: 5
Input Data: [A, C, T, W, null, C, D, W, null, null, A, C, T
Items: [A, C, D, T, W]
Individual Items Count: {A=4, C=6, D=4, T=4, W=5}
Individual Items Count(Sorted): {C=6, W=5, A=4, D=4, T=4}
Assigned prime number to items: {C=2, W=3, A=5, D=7, T=11}
Item Row count: [21, 12, 21, 17, 28, 20]


Individual Items calculated Value:
**********************************
Item: C : [19, 10, 19, 15, 26, 18]
Item: W : [16, 7, 16, 12, 23, 15]
Item: A : [11, 2, 11, 7, 18, 10]
Item: D : [9, 0, 9, 5, 16, 8]
Item: T : [5, -4, 5, 1, 12, 4]
Item: A : [11, 2, 11, 7, 18, 10]
Item: D : [4, -5, 4, 0, 11, 3]
Item: T : [0, -9, 0, -4, 7, -1]

Frequent Items:
***************
C -> W -> A -> D
C -> W -> A -> T
C -> W -> D
----------------------------------
CWD : 3 times
CWAD : 2 times
CWAT : 3 times
Maximum frequent items Count: 3

********** FREQUENT ITEMS **************
CWD:3
CWAT:3
****************************************
```

Figure 4: procedure to find maximal frequent itemsets

Complexity of Apriori algorithm depends mainly on the transaction itemsets. The complexity increases by 'n' when there are n items in the transaction i.e. the items that start from 1 frequent itemset till the n frequent itemsets.

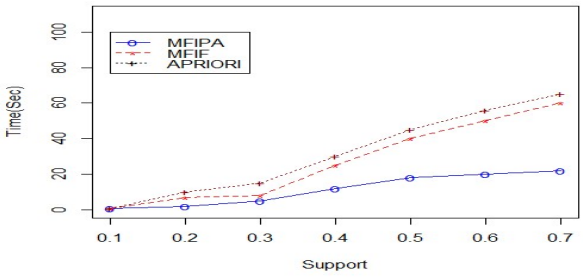

Figure 5:Graphical representation of time Complexity Comparison between MFIPA, Apriori and MFIF

PRIMA the time complexity is less when compared with Apriori here the top down approach is used so even though it leads to failure finding out all the subsets become must. MFIPA results in less time complexity compared to Apriori and MFIF when itemsets are large, it does not depends on the value 'n' complexity increases only at the generation of subsets of each itemsets, and yields less time complexity if maximal frequent itemset found at the initial stage.

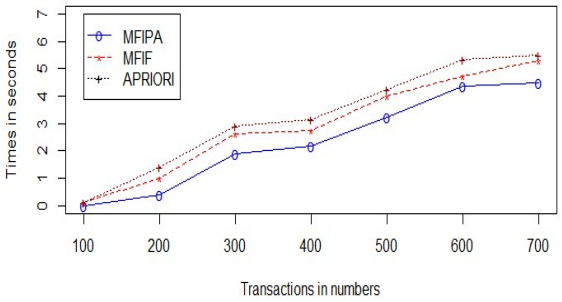

Figure 7: Performance of MFIPA on Market Basket dataset

Table 7. Comparison between MFIPA,MFIF, APRIORI

Transactions	MFIPA (time in seconds)	MFIF (time in seconds)	APRIORI (time in seconds)
100	0.006	0.016	0.187
500	0.050	0.062	0.422
5000	0.199	0.266	1.047

The results are shown in Table 1 and graphical comparison in Figure.5. Time taken by MFIPA, MFIF and Apriori for 100, 500, 5000 transactions is shown in the table.

The performance of MFIPA is compared with three different dataset. It is observed that the performance can vary significantly depending on the dataset characteristics. To evaluate the performance of MFIPA, three different benchmark datasets are used. All these datasets can be downloaded from UCI Machine Learning Repository [10].

Datasets taken for experiments are:

- Covid'19 dataset,
- Market Basket dataset,
- Heart dataset.

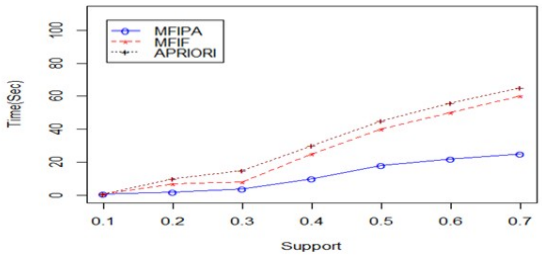

Figure 6: Performance of MFIPA on Covid'19 dataset

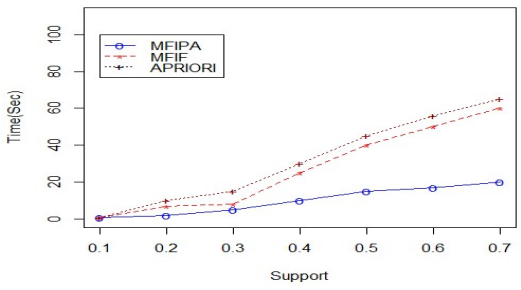

Figure 8: Performance of MFIPA on Heart dataset

Time complexity of MFIPA is less than MFIF and Apriori. Complexity of Apriori will increases as the number of items in the frequent itemset increases and MFIF complexity does not depend on the number of itemsets present. But the time complexity of MFIPA increases only at the time of subset generation.

6. ADVANTAGES

The main advantages of MFIPA method are as follows:

Too much memory space is not required for generation of subsets, because at a given time only one level of element subsets is generated; as shown above only 12 element subsets are generated.

Any element frequent set can be generated in a single scan by subset creation method, which will help in applying any search method to traverse and to get maximal frequent itemset, also it helps in reducing the scans drastically.

Assigning of prime numbers in inverse by proportional relationship to the item frequency which guarantees less value for prime multiplication and hence less storage and less computational complexity can be achieved. The prime method helps to find the order in which the data items must be scanned.

7. CONCLUSION

In data mining, learning association rule is a popular and well researched method for discovering interesting relations between objects in large databases. An efficient way to discover the maximal frequent is very important in some kinds of data mining problems .The maximal frequent set provides an effective representation of all the frequent itemsets. Discovering maximal frequent itemsets implies immediate discovery of all frequent itemsets. This paper presents a new algorithm that can efficiently discover the maximal frequent set in massive datasets. The top-down searching strategy is adopted in this algorithm. Substitution of prime value approach is very significant and effective to find maximal frequent itemset. The algorithm is enhanced in different aspects, which helps to improve the performance of the algorithm. By incorporating all the mentioned enhancements in the proposed systems an efficient and intelligent method for mining frequent patterns can be developed. In this paper, we revisited the frequent itemset mining problem and proposed a new algorithm for mining frequent itemsets by using prime values. Experimental results on large scale datasets show that MFIPA achieves good scalability.

REFRENCES

[1] Bharati, M., & Ramageri, M. (2010). "Data Mining Techniques And Applications." Indian Journal of Computer Science and Engineering , Vol. 1 No. 4 301-305.

[2] Fernando Berzal, J. C. (2000). "TBAR: An Efficeient Method For Association Rule Mining In Relational Databases." Data & Knowledge Engineering , 47 - 64.

[3] Rania Mkhinini Gahar, Olfa Arfaoui, Minyar Sassi Hidri, Nejib Ben Hadj-Alouane, "Parallelcharmax: An Effective Maximal Frequent Itemset Mining Algorithm Based On Mapreduce Framework" Ieee/Acs 14th

International Conference on Computer Systems and Applications,2017

[4] K. Sumathi, S. Kannan and K. Nagarajan, "Maximal Frequent Itemset Mining Using Breadth-First Search With Efficient Pruning" International Conference on Computer Networks and Communication Technologies, Springer Nature Singapore Pte Ltd. 2019.

[5] Lin, D., Kedem, Z.M.: "Pincer-Search: A New Algorithm For Discovering The Maximum Frequent Set." In: Proceedings of VI International Conference on Extending Database Technology (1998)

[6] Karam Gouda ,Mohammed J. Zaki "GENMAX: An Efficient Algorithm For Mining Maximal Frequent Itemsets, Data Mining and Knowledge Discovery", 11, 223–242, 2005_c 2005 Springer Science

[7] Don-Lin Yang, Ching-Ting Pan and Yeh-Ching Chung , "An Efficient Hash-Based Method For Discovering The Maximal Frequent Set," Computer Software and Applications Conference, 2001.COMPSAC 2001. 25th Annual International, vol., no., pp.511-516, 2001

[8] Qinghua Zou, Wesley W. Chu and Baojing Lu, "Smartminer: A Depth First Algorithm Guided By Tail Information For Mining Maximal Frequent Itemsets" Proceedings 2002 IEEE International Conference on Data Mining. ICDM 2002 , vol., no., pp.570 – 577, 2002

[9] Hua-Fu Li, Suh-Yin Lee and Man-Kwan Shan, "Online Mining (Recently) Maximal Frequent Itemsets Over Data Streams", Research Issues in Data Engineering: Stream Data Mining and Applications,2005. RIDE-SDMA 2005. 15th International Workshop on , vol., no., pp. 11-18, 3-4 April 2005 International journal of Data Mining & Knowledge Management Process (IJDKP) Vol.3, No.1, January 2013 38

[10] Https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/